Metode directe pentru sisteme de ecuații liniare

Eliminare gaussiană, descompunere LU, Cholesky

Radu T. Trîmbițaș

Universitatea "Babeș-Bolyai"

6 aprilie 2020

Rezolvarea sistemelor liniare

- În notație matricială, un sistem se scrie sub forma

$$Ax = b$$

  unde $A \in \mathbb{K}^{n\times n}$, $b \in \mathbb{K}^{n\times 1}$.
- Soluția $x = A^{-1}b$
- În majoritatea problemelor practice inversarea este nenecesară și nerecomandabilă
- Exemplu extrem, $n = 1$: $7x = 21$, cu soluția $x = \frac{21}{7} = 3$, o operație /
- Rezolvat prin inversare ne dă $x = 7^{-1} \cdot 21 = 0.1428571 \cdot 21 = 3.0000002$, două operații /,*
- Considerații similare se aplică și la sisteme cu mai multe ecuații și chiar la sisteme cu aceeași matrice A și membri drepți diferiți

Un exemplu numeric I

- Considerăm exemplul

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 6 \end{bmatrix}$$

- $0.3E_1 + E_2 \to E_2$, $-0.5E_1 + E_3 \to E_3$. Coeficientul 10 al lui x_1 se numește *pivot*, iar cantitățile -0.3 și 0.5 obținute prin împărțirea coeficienților lui x_1 din celelalte ecuații se numesc *multiplicatori*

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.1 \\ 2.5 \end{bmatrix}$$

Un exemplu numeric II

- Pasul 2, eliminarea lui x_2 din a treia ecuație. Pivotul, coeficientul lui x_2, -0.1 este mai mic decât alți coeficienți. Din acest motiv ecuațiile 2 și 3 trebuie interschimbate, operație numită *pivotare*. În acest caz nu este necesară, dar în general este crucială.

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & -0.1 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 2.5 \\ 6.1 \end{bmatrix}$$

- Pasul 3, $0.04E_2 + E_3 \to E_3$ (Cât ar fi multiplicatorul fără interschimbare?)

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 2.5 \\ 6.2 \end{bmatrix}$$

Un exemplu numeric III

- Ultima ecuație
$$6.2x_3 = 6.2$$
ne dă $x_3 = 1$.
- Înlocuind în a doua ecuație
$$2.5x_2 + 5 \cdot 1 = 2.5 \Longrightarrow x_2 = -1.$$
- Înlocuind în prima ecuație
$$10x_1 + (-7)(-1) = 7 \Longrightarrow x_1 = 0.$$
- Soluția este
$$x = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

Un exemplu numeric IV

- Verificare

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 6 \end{bmatrix}.$$

- Algoritmul poate fi exprimat mai compact în formă matricială, fie

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & -0.04 & 1 \end{bmatrix}, U = \begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{bmatrix}, P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Matricea L este matricea multiplicatorilor, U este matricea finală, P descrie pivotarea și

$$LU = PA$$

- Matricea originală poate fi exprimată ca produs de matrice cu o structură mai simplă.

Factorizare LU

- Algoritmul folosit aproape universal pentru rezolvarea sistemelor liniare este eliminarea gaussiană.
- Cercetările din perioada 1955-1965 au evidențiat aspecte ale EG netratate până atunci: alegerea pivoților și interpretarea corectă a efectului erorilor de rotunjire
- EG are două stadii, triunghiularizarea matricei inițiale și substituția inversă

Matrice de permutare și triunghiulare I

- O *matrice de permutare* se obține din matricea identică prin permutări de linii sau coloane. O astfel de matrice are exact un 1 pe fiecare linie și coloană și în rest 0.

$$P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Efect: PA permutare de linii, AP permutare de coloane
- MATLAB utilizează și vectori de permutare ca indici de linie sau coloane; fie $p = \begin{bmatrix} 4 & 1 & 3 & 2 \end{bmatrix}$, `P*A` și `A(p,:)` sunt echivalente, la fel `A*P` și `A(:,p)`. Notația vectorială este mai rapidă și utilizează mai puțină memorie.

Matrice de permutare și triunghiulare II

- Soluția sistemului $Px = b$, P matrice de permutare, este $x = P^T b$, adică o rearanjare a componentelor lui b.
- O matrice triunghiulară superior are toate elementele nenule deasupra diagonalei principale sau pe ea, adică $a_{ij} = 0$ dacă $i > j$.
- Analog se definesc și matricele triunghiulare inferior. La rezolvarea sistemelor liniare sunt importante și matricele triunghiulare inferior care au toate elementele de pe diagonala principală egale cu 1 *unit lower triangular matrices*.
- Sistemele liniare cu matricea triunghiulară sunt ușor de rezolvat în timp $\Theta(m^2)$.
- **Măsura complexității -Numărul de operații în virgulă flotantă**.

Algoritm pentru sistem triunghiular superior

```
function x = backsubst(U,b)
%BACKSUBST - backward substitution
%U - upper triangular matrix
%b - right hand side vector

n=length(b);
x=zeros(size(b));
for k=n:-1:1
    x(k)=(b(k)-U(k,k+1:n)*x(k+1:n))/U(k,k);
end
```

Algoritm pentru sistem triunghiular inferior

```
function x=forwardsubst(L,b)
%FORWARDSUBST - forward substitution
%L - lower triangular matrix
%b - right hand side vector

x=zeros(size(b));
n=length(b);
for k=1:n
    x(k)=(b(k)-L(k,1:k-1)*x(1:k-1))/L(k,k);
end
```

Descompunerea LU

- Transformă $A \in \mathbb{C}^{m\times m}$ într-o matrice triunghiulară superior, U scăzând multiplii de linii
- Fiecare L_i introduce zerouri sub diagonală în coloana i:

$$\underbrace{L_{m-1}\dots L_2 L_1}_{L^{-1}} A = U \Longrightarrow A = LU \text{ unde } L = L_1^{-1} L_2^{-1} \dots L_{m-1}^{-1}$$

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}} \xrightarrow{L_1} \underset{L_1A}{\begin{bmatrix} \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \times & \times & \times \end{bmatrix}} \xrightarrow{L_2} \underset{L_2L_1A}{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \mathbf{0} & \times & \times \\ & \mathbf{0} & \times & \times \end{bmatrix}} \xrightarrow{L_3} \underset{L_3L_2L_1A}{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & \mathbf{0} & \times \end{bmatrix}}$$

- "Triunghiularizare triunghiulară"

Matricele L_k

- La pasul k se elimină elementele de sub A_{kk}:

$$x_k = \left[\begin{array}{cccccc} x_{11} & \cdots & x_{kk} & x_{k+1,k} & \cdots & x_{mk} \end{array}\right]^T$$
$$L_k x_k = \left[\begin{array}{cccccc} x_{11} & \cdots & x_{kk} & 0 & \cdots & 0 \end{array}\right]^T$$

- Multiplicatorii $\ell_{jk} = x_{jk}/x_{kk}$ apar in L_k:

$$L_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -\ell_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -\ell_{mk} & & & 1 \end{bmatrix}$$

Construcția lui L

- Matricea L conține toți multiplicatorii într-o singură matrice (cu semne $+$)

$$L = L_1^{-1} L_2^{-1} \dots L_{m-1}^{-1} = \begin{bmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{m1} & \ell_{m2} & \cdots & \ell_{m,m-1} & 1 \end{bmatrix}$$

- Definim $\ell_k = [0, \dots, 0, \ell_{k+1,k}, \dots, \ell_{m,k}]^T$. Atunci $L_k = I - \ell_k e_k^*$.
  - Avem $L_k^{-1} = I + \ell_k e_k^*$, deoarece $e_k^* \ell_k = 0$ și
    $\left(I - \ell_k e_k^*\right)\left(I + \ell_k e_k^*\right) = I - \ell_k e_k^* \ell_k e_k^* = I$
  - De asemenea, $L_k^{-1} L_{k+1}^{-1} = I + \ell_k e_k^* + \ell_{k+1} e_{k+1}^*$, deoarece $e_k^* \ell_{k+1} = 0$ și
    $\left(I + \ell_k e_k^*\right)\left(I + \ell_{k+1} e_{k+1}^*\right) = I + \ell_k e_k^* + \ell_{k+1} e_{k+1}^* + \ell_k e_k^* \ell_{k+1} e_{k+1}^*$

Eliminare gaussiană fără pivotare

- Se factorizează $A \in \mathbb{C}^{m\times m}$ în $A = LU$

Eliminare gaussiană fără pivot

$U := A$; $L = I$;
for $k := 1$ **to** $m-1$ **do**
  for $j := k+1$ **to** m **do**
    $\ell_{jk} := u_{jk} / u_{kk}$;
    $u_{j,k:m} := u_{j,k:m} - \ell_{jk} u_{k,k:m}$;

- Ciclul interior poate fi scris utilizând operații matriciale în loc de cicluri **for**
- Număr de operații (complexitatea) $\sim \sum_{k=1}^{m} 2(m-k)(m-k) \sim 2\sum_{k=1}^{m} k^2 \sim \frac{2}{3}m^3$

Eliminare gaussiană cu produs exterior

- Ciclul interior poate fi scris cu operații matriciale în loc de for

Eliminare gaussiană cu produs exterior

for $k := 1$ **to** $m-1$ **do**
  $rows := k+1 : m$;
  $A_{rows,k} := A_{rows,k} / A_{k,k}$;
  $A_{rows,rows} := A_{rows,rows} - A_{rows,k} A_{k,rows}$;

Matricea $S = A_{k+1:m,k+1:m} - A_{k+1:m,k} A_{k,k+1:m}$ se numește complement Schur al lui A în raport cu $a_{k,k}$.

Necesitatea pivotării I

- EG așa cum a fost prezentată este instabilă.
- Pentru anumite matrice EG poate eșua, datorită tentativei de împărțire la zero
$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$
- Matricea este nesingulară și bine condiționată; $condA = \frac{3+\sqrt{5}}{2} \approx 2.168$
- Fenomenul este mai general; este evidențiat de o ușoară perturbație a lui A
$$A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$$

Necesitatea pivotării II

- Acum procesul nu eșuează; se obține (în aritmetica exactă)

$$L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}$$

- În aritmetica în virgulă flotantă, dublă precizie, $1 - 10^{20}$ nu este reprezentabil exact, el se va rotunji la -10^{20}
- Factorii calculați ai descompunerii vor fi

$$\widetilde{L} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \quad \widetilde{U} = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix}$$

- Produsul $\widetilde{L}\widetilde{U}$ nu este apropiat de A

$$\widetilde{L}\widetilde{U} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \cdot \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix}$$

Necesitatea pivotării III

- Rezolvând sistemul
$$\widetilde{L}\widetilde{U}x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
se obține $\widetilde{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, dar soluția corectă este $x = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$
- Explicație: *EG nu este nici regresiv stabilă, nici stabilă (ca algoritm de factorizare). Mai mult, matricele triunghiulare obținute pot fi foarte prost condiționate, introducându-se astfel o sursă suplimentară de instabilitate.*
- **Observație:** Dacă un pas al unui algoritm nu este regresiv stabil, algoritmul întreg poate fi instabil.

Pivotare

- La pasul k, am utilizat elementul k, k al matricei ca pivot și am introdus zerouri în coloana k a liniilor rămase

$$
\begin{bmatrix}
\times & \times & \times & \times & \times \\
 & \mathbf{x}_{kk} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\
 & \times & \times & \times & \times \\
 & \times & \times & \times & \times \\
 & \times & \times & \times & \times
\end{bmatrix}
\longrightarrow
\begin{bmatrix}
\times & \times & \times & \times & \times \\
 & x_{kk} & \times & \times & \times \\
 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\
 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\
 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times}
\end{bmatrix}
$$

- Dar, orice alt element $i \geq k$ din coloana k poate fi utilizat ca pivot:

$$
\begin{bmatrix}
\times & \times & \times & \times & \times \\
 & \times & \times & \times & \times \\
 & \times & \times & \times & \times \\
 & \mathbf{x}_{ik} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\
 & \times & \times & \times & \times
\end{bmatrix}
\longrightarrow
\begin{bmatrix}
\times & \times & \times & \times & \times \\
 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\
 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\
 & x_{ik} & \times & \times & \times \\
 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times}
\end{bmatrix}
$$

Pivotare

- De asemenea, se poate utiliza orice altă coloană $j \geq k$:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \boldsymbol{\times} & \mathbf{x}_{ij} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \times & \times & \times & \times \end{bmatrix} \longrightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ & \boldsymbol{\times} & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \times & x_{ij} & \times & \times \\ & \boldsymbol{\times} & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}$$

- Alegând diferiți pivoți ne asigurăm că putem evita pivoții nuli sau foarte mici
- În loc să utilizăm pivoți în poziții diferite, putem interschimba linii sau coloane și să utilizăm algoritmul standard (pivotare)
- O implementare concretă poate face pivotarea indirect, fără a muta fizic datele

Pivotare parțială

- Alegerea pivoților dintre toți candidații valizi este costisitoare(pivotare completă)
- Considerăm doar pivoții din coloana k și interschimbăm liniile(pivotare parțială)

$$
\underset{\text{Selecție pivot}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \mathbf{x}_{ik} & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}}
\xrightarrow{P_1}
\underset{\text{Interschimbare linii}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \mathbf{x}_{ik} & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}}
\xrightarrow{L_1}
\underset{\text{Eliminare}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & x_{ik} & \times & \times & \times \\ & \mathbf{0} & \times & \times & \times \\ & \mathbf{0} & \times & \times & \times \\ & \mathbf{0} & \times & \times & \times \end{bmatrix}}
$$

- Cu operații matriceale:

$$L_{m-1}P_{m-1}\ldots L_2P_2L_1P_1A = U$$

Factorizarea PA = LU

- Pentru a combina toți L_k și toți P_k în forma dorită de noi, rescriem factorizarea precedentă sub forma

$$L_{m-1}P_{m-1}\dots L_2P_2L_1P_1A = U$$
$$\left(L'_m\cdots L'_2L'_1\right)\left(P_{m-1}\cdots P_2P_1\right)A = U$$

unde

$$L'_k = P_{m-1}\cdots P_{k+1}L_kP_{k+1}^{-1}\cdots P_{m-1}^{-1}$$

- Aceasta ne dă factorizare (descompunerea) LU a lui A

$$PA = LU$$

Eliminarea gaussiană cu pivotare parțială

- Factorizează $A \in \mathbb{C}^{m\times m}$ în $PA = LU$

Eliminare gaussiană cu pivotare parțială

$U := A$; $L := I$; $P := I$;
for $k := 1$ **to** $m-1$ **do**
  Alege $i \geq k$ care maximizeaza $|u_{ik}|$;
  $u_{k,k:m} \leftrightarrow u_{i,k:m}$; {interschimbare}
  $\ell_{k,1:k-1} \leftrightarrow \ell_{i,1:k-1}$;
  $P_{k,:} \leftrightarrow P_{i,:}$;
  for $j := k+1$ **to** m **do**
    $\ell_{jk} := u_{jk}/u_{kk}$;
    $u_{j,k:m} := u_{j,k:m} - \ell_{jk}u_{k,k:m}$;

Algoritmul devine mai eficient dacă se fac toate calculele în situ (în matricea A).

Cod MATLAB pentru descompunerea LUP

```
function [L,U,P]=lup(A)
%LUP - LUP decomposition of A
%permute effectively lines

[m,n]=size(A);
P=zeros(m,n);
piv=(1:m)';
for i=1:m-1
    %pivoting
    [pm,kp]=max(abs(A(i:m,i)));
    kp=kp+i-1;
    %line interchange
    if i~=kp
        A([i,kp],:)=A([kp,i],:);
        piv([i,kp])=piv([kp,i]);
    end
    %Schur complement
    lin=i+1:m;
    A(lin,i)=A(lin,i)/A(i,i);
    A(lin,lin)=A(lin,lin)-...
     A(lin,i)*A(i,lin);
end;
for i=1:m
    P(i,piv(i))=1;
end;
U=triu(A);
L=tril(A,-1)+eye(m);
```

Exemplu

- Rezolvați sistemul

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 4 & 2 \end{bmatrix} x = \begin{bmatrix} 3 \\ 4 \\ 8 \end{bmatrix}$$

prin descompunere LUP.

- **Soluție**: Avem

$$\left[\begin{array}{c|ccc} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 2 \\ 3 & 2 & 4 & 2 \end{array}\right] \sim \left[\begin{array}{c|ccc} 3 & 2 & 4 & 2 \\ 2 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{array}\right] \sim \left[\begin{array}{c|ccc} 3 & 2 & 4 & 2 \\ 2 & \frac{1}{2} & 1 & 2 \\ 1 & \frac{1}{2} & 1 & 1 \end{array}\right]$$

$$\left[\begin{array}{c|ccc} 3 & 2 & 4 & 2 \\ 2 & \frac{1}{2} & -1 & 1 \\ 1 & \frac{1}{2} & -1 & 0 \end{array}\right] \sim \left[\begin{array}{c|ccc} 3 & 2 & 4 & 2 \\ 2 & \frac{1}{2} & -1 & 1 \\ 1 & \frac{1}{2} & 1 & 0 \end{array}\right] \sim \left[\begin{array}{c|ccc} 3 & 2 & 4 & 2 \\ 2 & \frac{1}{2} & -1 & 1 \\ 1 & \frac{1}{2} & 1 & -1 \end{array}\right].$$

Exemplu (continuare)

- Deci

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & 4 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

- Sistemele triunghiulare corespunzătoare sunt

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 1 & 1 \end{bmatrix} y = Pb = \begin{bmatrix} 8 \\ 4 \\ 3 \end{bmatrix},$$

cu soluția $y = [8, 0, -1]^T$

Exemplu (continuare)

- și

$$\begin{bmatrix} 2 & 4 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} x = \begin{bmatrix} 8 \\ 0 \\ -1 \end{bmatrix},$$

cu soluția $x = [1, 1, 1]^T$.

- Verificare

$$PA = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 4 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 2 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$LU = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 4 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 2 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}.$$

Pivotare totală

- Dacă se selectează pivoți din coloane diferite, sunt necesare matrice de permutare la stânga Q_k:

$$\begin{aligned} L_{m-1}P_{m-1}\cdots L_2P_2L_1P_1AQ_1Q_2\cdots Q_{m-1} &= U \\ (L'_{m-1}\cdots L'_2L'_1)(P_{m-1}\cdots P_2P_1)A(Q_1Q_2\cdots Q_{m-1}) &= U \end{aligned}$$

- Punem

$$\begin{aligned} L &= (L'_{m-1}\cdots L'_2L'_1)^{-1} \\ P &= P_{m-1}\cdots P_2P_1 \\ Q &= Q_1Q_2\cdots Q_{m-1} \end{aligned}$$

pentru a obține

$$PAQ = LU$$

Liu Hui c. 220 –c. 280
Matematician chinez, a discutat eliminarea ,,gaussiană" în comentariile sale asupra lucrării ,,Cele nouă capitole ale artei matematice" 263 AD

Carl Friedrich Gauss 1777-1855
Matematică, astronomie, geodezie, magnetism
1809 GE
(Ca adolescent în Braunschweig a descoperit teorema binomială, reciprocitatea pătratică, media aritmetico-geometrică. . .)
1807-1855: Universitatea din Göttingen

Stabilitatea LU fără pivotare

- Pentru $A = LU$ calculată fără pivotare:

$$\widetilde{L}\widetilde{U} = A + \delta A, \qquad \frac{\|\delta A\|}{\|L\|\,\|U\|} = O(\mathrm{eps})$$

- Eroare se referă la $\widetilde{L}\widetilde{U}$, nu la $\widetilde{L}$ sau $\widetilde{U}$
- Notă: la numitor apare $\|L\|\,\|U\|$, nu $\|A\|$
- $\|L\|$ și $\|U\|$ pot fi arbitrar de mari, de exemplu

$$A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}$$

- Deci, algoritmul este nestabil

Stabilitatea LU cu pivotare

- Daca se face pivotare, toate elementele lui L sunt ≤ 1 în modul, deci $\|L\| = O(1)$
- Pentru a măsura creșterea lui U, se introduce factorul de creștere

$$\rho = \frac{\max_{ij} |u_{ij}|}{\max_{ij} |a_{ij}|}$$

care implică $\|U\| = O(\rho \|A\|)$

- Pentru descompunerea $PA = LU$ calculată cu pivotare:

$$\widetilde{L}\widetilde{U} = PA + \delta A, \qquad \frac{\|\delta A\|}{\|A\|} = O(\rho \mathrm{eps})$$

- Dacă $\rho = O(1)$, atunci algoritmul este regresiv stabil

Factorul de creștere I

- Considerăm matricea

$$\begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & 1 & & \\ -1 & -1 & -1 & 1 & \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & 1 & 8 \\ & & & & 16 \end{bmatrix}$$

- Nu apare nici o pivotare, deci aceasta este o factorizare $PA = LU$
- Factorul de creștere $\rho = 16 = 2^{m-1}$ (se poate arăta că acesta este cazul cel mai nefavorabil)
- Deci, $\rho = 2^{m-1} = O(1)$, uniform, pentru toate matricele de dimensiune m
- Regresiv stabil conform definiției, dar rezultatul poate fi inutil
- Totuși, nu se știe exact de ce, factorii de creștere sunt mici în practică

Factorul de creștere II

- **Conjectură**: factorii de creștere de ordin mai mare ca $1/2$ sunt rari în practică
- adică, pentru orice $\alpha > 1/2$ și $M > 0$, probabilitatea evenimentului $\rho > m^{\alpha}$ este mai mică decât m^{-M}, pentru m suficient de mare.

$$\forall \alpha > \frac{1}{2} \forall M > 0 \exists m_0, \forall m > m_0 \; P(\rho > m^{\alpha}) < m^{-M}.$$

- Problemă deschisă: conjectura este adevărată sau falsă?

Matrice SPD

- Reamintim:
  - $A \in \mathbb{R}^{m\times m}$ este simetrică dacă $a_{ij} = a_{ji}$, sau $A = A^T$
  - $A \in \mathbb{C}^{m\times m}$ este hermitiană dacă $a_{ij} = \bar{a}_{ji}$, sau $A = A^*$
- O matrice hermitiană A este hermitian pozitiv definită dacă $x^*Ax > 0$ pentru $x \neq 0$
  - x^*Ax este întotdeauna real deoarece $x^*Ay = \overline{y^*Ax}$
  - Simetric pozitiv definită, sau SPD, pentru matrice reale
- dacă A este $m \times m$ PD și X are rang maxim, atunci X^*AX este PD
  - Deoarece $(X^*AX)^* = X^*AX$, și dacă $x \neq 0$ atunci $Xx \neq 0$ și $x^*(X^*AX)x = (Xx)^*A(Xx) > 0$
  - Orice submatrice principală a lui A este PD, și orice element diagonal $a_{ii} > 0$
- matricele PD au valori proprii reale pozitive și vectori proprii ortonormali

Factorizarea Cholesky

- Se elimină sub pivot și la dreapta pivotului (datorită simetriei):

$$A = \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ w/\alpha & I \end{bmatrix} \begin{bmatrix} \alpha & w^*/\alpha \\ 0 & K - ww^*/a_{11} \end{bmatrix}$$
$$= \begin{bmatrix} \alpha & 0 \\ w/\alpha & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & K - ww^*/a_{11} \end{bmatrix} \begin{bmatrix} \alpha & w^*/\alpha \\ 0 & I \end{bmatrix} = R_1^* A_1 R_1$$

unde $\alpha = \sqrt{a_{11}}$

- $K - ww^*/a_{11}$ este o submatrice principală a matricei PD $R_1^{*-1} A R_1^{-1}$, deci *elementul ei din stânga sus este pozitiv*

Factorizarea Cholesky

- Se aplică recursiv și se obține

$$A = (R_1^* R_2^* \dots R_m^*)(R_m \dots R_2 R_1) = R^* R, \quad r_{ii} > 0$$

- Existența și unicitatea: orice matrice HPD are o factorizare Cholesky unică
  - Algoritmul recursiv de pe slide-ul precedent nu eșuează niciodată
  - Rezultă și unicitatea, deoarece $\alpha = \sqrt{a_{11}}$ este determinat unic (dat) la fiecare pas și la fel, întreaga linie w/α

Algoritmul de factorizare Cholesky

- Factorizează matricea HPD $A \in \mathbb{C}^{m\times m}$ în $A = R^*R$:

Factorizare Cholesky

$R := A$;
for $k := 1$ **to** m **do**
  for $j := k+1$ **to** m **do**
    $R_{j,j:m} := R_{j,j:m} - R_{k,j:m}\overline{R_{k,j}}/R_{k,k}$
  $R_{k,k:m} := R_{k,k:m}/\sqrt{R_{k,k}}$

- Complexitatea (număr de operații)

$$\sum_{k=1}^{m}\sum_{j=k+1}^{m} 2(m-j) \sim 2\sum_{k=1}^{m}\sum_{j=1}^{k} j \sim \sum_{k=1}^{m} k^2 \sim \frac{m^3}{3}$$

Exemplu

- Să se rezolve sistemul

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 3 & 3 \end{bmatrix} x = \begin{bmatrix} 4 \\ 10 \\ 7 \end{bmatrix}$$

folosind descompunerea Cholesky.

- **Soluție**: Calculând radicalii pivoților și complementele Schur se obține

$$B = \begin{bmatrix} 1 & 2 & 1 \\ & 5 & 3 \\ & & 3 \end{bmatrix} \sim \begin{bmatrix} 1 & 2 & 1 \\ & 1 & 1 \\ & & 2 \end{bmatrix} \sim \begin{bmatrix} 1 & 2 & 1 \\ & 1 & 1 \\ & & 1 \end{bmatrix}.$$

Exemplu

- Sistemele corespunzătoare sunt:

$$\begin{bmatrix} 1 & & \\ 2 & 1 & \\ 1 & 1 & 1 \end{bmatrix} y = \begin{bmatrix} 4 \\ 10 \\ 7 \end{bmatrix}$$

  cu soluția $y = \begin{bmatrix} 4 & 2 & 1 \end{bmatrix}^T$

- și

$$\begin{bmatrix} 1 & 2 & 1 \\ & 1 & 1 \\ & & 1 \end{bmatrix} x = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix},$$

  cu soluția $x = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$.

Stabilitatea

- Factorul Cholesky calculat $\widetilde{R}$ satisface

$$\widetilde{R}^*\widetilde{R} = A + \delta A, \qquad \frac{\|\delta A\|}{\|A\|} = O(\mathrm{eps})$$

  algoritmul este regresiv stabil
- Dar, eroarea în $\widetilde{R}$ poate fi mare ,

$$\left\|\widetilde{R} - R\right\| / \|R\| = O(\kappa(A)\mathrm{eps})$$

- Rezolvare $Ax = b$ pentru HPD A și cu două substituții
  - Numărul de operații Cholesky $\sim m^3/3$
  - Algoritm regresiv stabil:

$$(A + \Delta A)\widetilde{x} = b, \qquad \frac{\|\Delta A\|}{\|A\|} = O(\mathrm{eps})$$

John von Neumann (1903-1957)

André Louis Cholesky (1875-1918)

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă
2. Dacă `A` este o permutare a unei matrice triunghiulare, se rezolvă prin substituție (utilă pentru `[L,U]=lu(A)` căci `L` este permutată)

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă
2. Dacă `A` este o permutare a unei matrice triunghiulare, se rezolvă prin substituție (utilă pentru `[L,U]=lu(A)` căci `L` este permutată)
3. Dacă `A` este simetrică sau hermitiană

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă
2. Dacă `A` este o permutare a unei matrice triunghiulare, se rezolvă prin substituție (utilă pentru `[L,U]=lu(A)` căci `L` este permutată)
3. Dacă `A` este simetrică sau hermitiană
   - Se verifică dacă toate elementele diagonale sunt pozitive

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă
2. Dacă `A` este o permutare a unei matrice triunghiulare, se rezolvă prin substituție (utilă pentru `[L,U]=lu(A)` căci `L` este permutată)
3. Dacă `A` este simetrică sau hermitiană
    - Se verifică dacă toate elementele diagonale sunt pozitive
    - Se încearca cu Cholesky; daca se termina cu succes se rezolvă prin substituție

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă
2. Dacă `A` este o permutare a unei matrice triunghiulare, se rezolvă prin substituție (utilă pentru `[L,U]=lu(A)` căci `L` este permutată)
3. Dacă `A` este simetrică sau hermitiană
   - Se verifică dacă toate elementele diagonale sunt pozitive
   - Se încearca cu Cholesky; daca se termina cu succes se rezolvă prin substituție
4. Dacă `A` este Hessenberg, se reduce la o matrice triunghiulară superior și apoi se rezolvă prin substituție inversă

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă
2. Dacă `A` este o permutare a unei matrice triunghiulare, se rezolvă prin substituție (utilă pentru `[L,U]=lu(A)` căci `L` este permutată)
3. Dacă `A` este simetrică sau hermitiană
   - Se verifică dacă toate elementele diagonale sunt pozitive
   - Se încearca cu Cholesky; daca se termina cu succes se rezolvă prin substituție
4. Dacă `A` este Hessenberg, se reduce la o matrice triunghiulară superior și apoi se rezolvă prin substituție inversă
5. Dacă `A` este pătratică, se factorizează `PA = LU` și se rezolvă prin substituție inversă

Backslash în MATLAB

- `x=A\b` pentru `A` densă realizează următorii pași

1. Dacă `A` este triunghiulară superior sau inferior se rezolvă prin substituție inversă sau directă
2. Dacă `A` este o permutare a unei matrice triunghiulare, se rezolvă prin substituție (utilă pentru `[L,U]=lu(A)` căci `L` este permutată)
3. Dacă `A` este simetrică sau hermitiană
   - Se verifică dacă toate elementele diagonale sunt pozitive
   - Se încearca cu Cholesky; daca se termina cu succes se rezolvă prin substituție
4. Dacă `A` este Hessenberg, se reduce la o matrice triunghiulară superior și apoi se rezolvă prin substituție inversă
5. Dacă `A` este pătratică, se factorizează `PA = LU` și se rezolvă prin substituție inversă
6. Dacă `A` nu este pătratică, se face factorizare QR cu metoda Householder, și se rezolvă problema de aproximare în sensul celor mai mici pătrate

Descompunere QR

- Fie $A \in \mathbb{C}^{m\times n}$. Se numește descompunere QR a lui A perechea de matrice (Q, R) unde $Q \in \mathbb{C}^{m\times n}$ este unitară, $R \in \mathbb{C}^{n\times n}$ este triunghiulară superior și $A = QR$.

Triunghiularizare Householder

- Metoda lui Householder înmulțește cu matrice unitare pentru a transform matricea într-una triunghiulară; de exemplu la primul pas:

$$Q_1 A = \begin{bmatrix} r_{11} & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}$$

- La sfârșit, am obținut un produs de matrice ortogonale

$$\underbrace{Q_n \dots Q_2 Q_1}_{Q^*} A = R$$

- "Triunghiularizare ortogonală"

Introducerea de zerouri

- Q_k introduce zerouri sub diagonală în coloana k
- Păstrează zerourile introduse anterior

$$\underset{A}{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1} \underset{Q_1 A}{\begin{bmatrix} \times & \times & \times \\ \mathbf{0} & \times & \times \\ \mathbf{0} & \times & \times \\ \mathbf{0} & \times & \times \\ \mathbf{0} & \times & \times \end{bmatrix}} \xrightarrow{Q_2} \underset{Q_2 Q_1 A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & \mathbf{0} & \times \\ & \mathbf{0} & \times \\ & \mathbf{0} & \times \end{bmatrix}} \xrightarrow{Q3} \underset{Q_3 Q_2 Q_1 A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \\ & & \mathbf{0} \\ & & \mathbf{0} \end{bmatrix}}$$

Reflectori Householder

- Fie Q_k de forma

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}$$

  unde I este $(k-1) \times (k-1)$ și F este $(m-k+1) \times (m-k+1)$

- Creăm reflectorul Householder F care introduce zerouri:

$$x = \begin{bmatrix} \times \\ \times \\ \vdots \\ \times \end{bmatrix} \qquad Fx = \begin{bmatrix} \|x\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \|x\| \, e_1$$

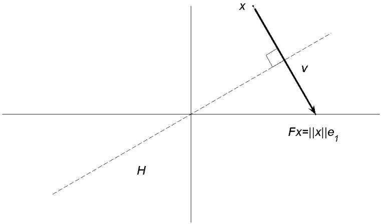
Reflectori Householder-Ideea

- Ideea: reflectăm în raport cu hiperplanul H, ortogonal pe $v = \|x\|_2 e_1 - x$, aplicând matricea unitară

$$F = I - 2\frac{vv^*}{v^*v}$$

- A se compara cu proiectorul

$$P_{\perp v} = I - \frac{vv^*}{v^*v}$$



Determinarea reflectorului

- reflexie Householder: $P = I - 2uu^T$, $\|u\|_2 = 1$; P simetrică și ortogonală, deoarece $P = P^T$ și

$$PP^T = \left(I - 2uu^T\right)\left(I - 2uu^T\right) = I - 4uu^T + 4uu^Tuu^T = I$$

- Dorim $Px = [c, 0, \dots, 0]^T = ce_1$ (anulăm toate componentele lui x exceptând prima)

$$Px = x - 2u(u^Tx) = ce_1 \Longrightarrow u = \frac{1}{2u^Tx}(x - ce_1)$$
$$\|x\|_2 = \|Px\|_2 = |c|$$

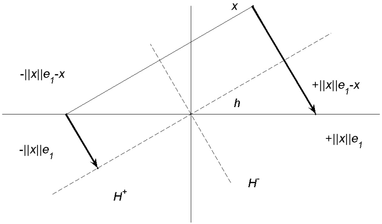
- obținem u paralel cu $\widetilde{u} = x \pm \|x\|_2 e_1$, deci $u = \widetilde{u} / \|\widetilde{u}\|_2$. Orice alegere de semn corespunde; vom alege

$$\widetilde{u} = [x_1 + sign(x_1)\|x\|_2, x_2, \dots, x_n]^T, \qquad u = \widetilde{u} / \|\widetilde{u}\|_2 .$$

Alegerea reflectorului

- Putem aplica reflexia oricărui multiplu z al lui $\|x\| e_1$ cu $|z| = 1$
- Proprietăți numerice mai bune pentru $\|v\|$ mare, de exemplu $v = \operatorname{sign}(x_1) \|x\| e_1 + x$

  - Notă: $\operatorname{sign}(0) = 1$, dar în MATLAB, `sign(0)==0`



Algoritmul lui Householder

- Calculează factorul R al descompunerii QR a matricei $m \times n$ A ($m \geq n$)
- Lasă rezultatul în A, memorând vectorii de reflexie v_k pentru utilizare ulterioară

Factorizare QR prin metoda Householder

for $k := 1$ **to** n **do**

$x := A_{k:m,k}$;

$v_k := \operatorname{sign}(x_1)\|x\|_2 e_1 + x$;

$v_k := v_k / \|v_k\|_2$;

$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k\left(v_k^* A_{k:m,k:n}\right)$

Aplicarea sau obținerea lui Q

- Calculăm $Q^* b = Q_n \dots Q_2 Q_1 b$ și $Qx = Q_1 Q_2 \dots Q_n x$ implicit
- Pentru a crea Q explicit, aplicăm pentru $x = I$

Calculul implicit al lui $Q^* b$

for $k := 1$ **to** n **do**

$$b_{k:m} = b_{k:m} - 2v_k \left(v_k^* b_{k:m} \right);$$

Calculul implicit al lui Qx

for $k := n$ **downto** 1 **do**

$$x_{k:m} = x_{k:m} - 2v_k \left(v_k^* x_{k:m} \right);$$

Complexitatea QR-Householder

- Cea mai mare parte a efortului

$$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k \left(v_k^* A_{k:m,k:n}\right)$$

- Operații pe iterație:
  - $2(m-k)(n-k)$ pentru produsele scalare $v_k^* A_{k:m,k:n}$
  - $(m-k)(n-k)$ pentru produsul exterior $2v_k(\cdots)$
  - $(m-k)(n-k)$ pentru scăderea $A_{k:m,k:n} - \cdots$
  - $4(m-k)(n-k)$ total
- Incluzând ciclul exterior, totalul devine

$$\sum_{k=1}^{n} 4(m-k)(n-k) = 4\sum_{k=1}^{n}\left(mn - k\left(m+n\right) + k^2\right)$$

$$\sim 4mn^2 - 4(m+n)n^2/2 + 4n^3/3 = 2mn^2 - 2n^3/3$$

Figura: Alston S. Householder (1904-1993), matematician american. Contribuții importante: biologie matematică, algebră liniară numerică. Cartea sa "The Theory of Matrices in Numerical Analysis" a avut un mare impact asupra dezvoltării analizei numerice și a informaticii.

Figura: James Wallace Givens (1910-1993) Pionier al algebrei liniare numerice și informaticii

Exemplu

Calculați descompunerea QR a matricei

$$A = \left[\begin{array}{cc} 3 & 1 \\ 4 & 1 \end{array} \right].$$

Soluție. Reflexia pentru prima coloană este $P = I - 2uu^T$. Vectorul u se determină astfel:

$$\begin{aligned}
\widetilde{u} &= \left[\begin{array}{c} x_1 + sign(x_1)\left\|x\right\|_2 \\ x_2 \end{array} \right] = \left[\begin{array}{c} 3+5 \\ 4 \end{array} \right] = \left[\begin{array}{c} 8 \\ 4 \end{array} \right]. \\
\left\|\widetilde{u}\right\| &= \sqrt{8^2+4^2} = 4\sqrt{5} \\
u &= \frac{\widetilde{u}}{\left\|\widetilde{u}\right\|} = \left[\begin{array}{c} \frac{2\sqrt{5}}{5} \\ \frac{\sqrt{5}}{5} \end{array} \right].
\end{aligned}$$

Matricea de reflexie este

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - 2 \begin{bmatrix} \frac{2\sqrt{5}}{5} \\ \frac{\sqrt{5}}{5} \end{bmatrix} \begin{bmatrix} \frac{2\sqrt{5}}{5} \\ \frac{\sqrt{5}}{5} \end{bmatrix}^T$$

$$= \begin{bmatrix} -\frac{3}{5} & -\frac{4}{5} \\ -\frac{4}{5} & \frac{3}{5} \end{bmatrix} = Q^T.$$

Se obține:

$$Q = \begin{bmatrix} -\frac{3}{5} & -\frac{4}{5} \\ -\frac{4}{5} & \frac{3}{5} \end{bmatrix}$$

$$R = Q^T A = \begin{bmatrix} -\frac{3}{5} & -\frac{4}{5} \\ -\frac{4}{5} & \frac{3}{5} \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 \\ 4 & 1 \end{bmatrix} = \begin{bmatrix} -5 & -\frac{7}{5} \\ 0 & -\frac{1}{5} \end{bmatrix}.$$

■

Rotații Givens I

- O rotație Givens

$$R(\theta)=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$$

roteşte un vector $x\in\mathbb{R}^2$ în sens trigonometric cu unghiul θ

- Rotația Givens cu unghiul θ în coordonatele i și j se obține cu ajutorul matricei de mai sus, punând elementele ei în liniile și coloanele i și j și în rest elementele matricei unitate.

Rotații Givens II

$$
R(i,j,\theta) := \begin{array}{c c}
 & \begin{array}{cccccccc} & & & i & & j & & \end{array} \\
\begin{array}{c} \\ \\ \\ i \\ \\ j \\ \\ \\ \\ \end{array} &
\begin{pmatrix}
1 & & & & & & & \\
 & 1 & & & & & & \\
 & & \ddots & & & & & \\
 & & & \cos\theta & & \sin\theta & & \\
 & & & & \ddots & & & \\
 & & & -\sin\theta & & \cos\theta & & \\
 & & & & & & \ddots & \\
 & & & & & & & 1 \\
 & & & & & & & & 1
\end{pmatrix}
\end{array}
$$

Rotații Givens III

- Dându-se x, i și j putem anula x_j alegând $\cos\theta$ și $\sin\theta$ astfel încât

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix},$$

adică

$$\cos\theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \qquad \sin\theta = \frac{x_j}{\sqrt{x_i^2 + x_j^2}}.$$

- Algoritmul QR bazat pe rotații Givens este analog algoritmului bazat pe reflexii Householder, dar când anulăm coloana i se anulează un element la un moment dat.